Discrete Flower Pollination Algorithm for Resource Constrained Project Scheduling Problem

Kirils Bibiks, Jian-Ping Li, Fun Hu
Faculty of Engineering and Informatics
University of Bradford
Bradford, United Kingdom

***Abstract*- In this paper, a new population-based and nature-inspired metaheuristic algorithm, Discrete Flower Pollination Algorithm (DFPA), is presented to solve the Resource Constrained Project Scheduling Problem (RCPSP). The DFPA is a modification of existing Flower Pollination Algorithm adapted for solving combinatorial optimization problems by changing some of the algorithm's core concepts, such as flower, global pollination, Lévy flight, local pollination. The proposed DFPA is then tested on sets of benchmark instances and its performance is compared against other existing metaheuristic algorithms. The numerical results have shown that the proposed algorithm is efficient and outperforms several other popular metaheuristic algorithms, both in terms of quality of the results and execution time. Being discrete, the proposed algorithm can be used to solve any other combinatorial optimization problems.**

Keywords- Flower Pollination Algorithm; Discrete Flower Pollination Algorithm; Combinatorial optimization; Resource Constrained Project Scheduling Problem; Evolutionary Computing.

I. Introduction

Resource Constrained Project Scheduling Problem (RCPSP) consists of a set of predefined tasks and resources and its main objective is to assign tasks to resources in such way, that overall project schedule is as cheap and short as possible. To make the schedule feasible, there are constraints that need to be satisfied.

Despite the simplicity of definition, RCPSP is one of the widely described combinatorial problems in the literature and has existed for at least 50 years [1]. Blazewicz et al. [1] describes RCPSP as a generalization of classical job-shop scheduling problem which belongs to the class of NP-hard optimization problems [2]. Kolisch [3] classified methods used for solving RCPCP as exact solution [4], Priority Rules-Based (PRB) [5] and metaheuristic approaches [6-8].

Exact methods guarantee to find an optimal solution if it exists. The most common exact method is the branch and bound algorithm [4, 10-11]. In the branch and bound algorithm a tree is generated, where each node represents a task. Sprecher and Drexl [12] claimed that those methods cannot be used to solve large scale problems, as the trees increase sharply with the increase of dimension sizes.

PRB methods employ one or more schemes to construct a feasible schedule. Panwalker and Iskander [5] surveyed a range of priority rules. Davis and Patterson [13] compared standard priority rules on a set of single-mode RCPSP and demonstrated that the heuristics' performance decreases when the constraints become too tight. After examining the most common priority rules, Browning [14] presented novel heuristics, based on tasks criticality and load balancing factors, which appeared to be more suitable for solving RCPSP. Lawrence and Morton [15] described priority rules by using a combination of project-, activity-, and resource-related metrics. Hildum [16] proposed priority rules that distinguish single- and multiple-priority rules approaches and outlined that a scheduler with

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

multiple priority rules shows better performance. Boctor [17] also had similar observations. Comparing with the exact solution methods, PBR methods can find solution in shorter time, however they cannot acquire global solution.

In the last decades the metaheuristic evolution-based computational methods have been getting a lot of attention and been used extensively to solve RCPSP. The metaheuristic methods start with initial solution and constantly improve it by successively executing operations which transform one or several solutions into others. There are many evolution-based metaheuristic methods, such Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and so on.

Husbands [18] outlined the advances of GA for scheduling and illustrated the resemblance between scheduling and sequence-based problems. Davis [19] demonstrated the benefits of using a stochastic search. Hartmann [6] proposed another implementation of GA and suggested to use a GA variation where every gene composing a chromosome is a delivery rule. Mendes et al. [20] proposed to use the priority rules to represent chromosomes in a form of a list of priority values for all activities in the project. Montoya-Torres [21] used a multi-array object-oriented model to depict chromosomes. Shahsavar et al. [22] designed a GA using a three-stage process that utilizes design of experiments and response surface methodology. Alcaraz et al. [23] developed several new variations of GA for solving RCPSP, extending the representation and operator previously designed for the single-mode version of the problem.

Aarts et al. [24] described one of the first SA approaches for scheduling problems. Palmer [25] combined planning and scheduling in a digraph representation. Boctor [26] reported fairly good performances of SA approaches on Patterson problems. Nikulin and Drexl [27] used SA to solve an airport flight gate scheduling problem which was modelled as RCPSP. Bouleimen [28] proposed that the conventional SA search scheme is replaced by a new design that takes into account the specificity of the solution space of the project scheduling problems. Zamani [29] combined a SA and time-windowing process, where SA generates an activities schedule and time-windowing improves it.

PSO is another popular metaheuristic method. Zhang [30] demonstrated good performance of PSO in solving RCPSP. Anantathanvit and Munlin [31] extended the original PSO algorithm by regrouping agent particles within the appropriate radius of circle. Li [32] replaced the complicated updating equations of the traditional PSO with one GA crossover operation to make the process quicker and less resource demanding. Linyi [33] introduced an implementation of PSO with one-point crossover for RCPSP. Zhang et al. [34] developed a variation of PSO in which the activities sequence is encoded with a simple code rule by the code orderer.

One of the first suggested uses of ACO for RCPSP was made by Merkle [35]. An improved ACO approach for solving RCPSP was introduced by Luo [36]. Wang [37] embedded a project priority indicator into ACO as the heuristic function and solved the multi-project scheduling problem. Shou [38] used an ACO with two separate ant colonies employed, where forward scheduling technique is applied by first ant colony, while backward scheduling technique is applied by the second one. The modified ACO algorithm for precedence and resource-constrained scheduling problems was presented by Lo et al. [39].

More and more approaches for solving RCPSP are being proposed in the literature. Recently, a new nature-inspired metaheuristic method called Flower Pollination Algorithm (FPA) has been developed by Yang [40]. Based

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

on the work done in [40], the FPA has demonstrated to be a very efficient algorithm in finding global optima with high success rates. Yang [40] showed that FPA is superior to both PSO and GA in terms of efficiency and success rate. However, since the FPA was designed for solving the continuous optimization problems, in order to apply it for RCPSP, the algorithm's core logic needs to be changed. The aim of this paper is to present a modification of the original FPA called Discrete Flower Pollination Algorithm (DFPA) which was adapted for solving the combinatorial problems.

The subsequent parts of this paper are organized as follows: The mathematical formulation of the problem is outlined in Section II; The explanation of FPA is given in Section III; The modification of FPA for RCPSP is proposed in Section IV; Simulation results and performance comparison with other popular algorithms are detailed in Section V; Finally, the conclusions and plans for future work are outlined in Section VI.

II. Mathematical Formulation of the Problem

The main objective of the RCPSP is to find optimal schedule with minimal duration by assigning a start time to each activity, with the precedence relations and the resource availabilities taken into account.

Activities are formalized by a finite set $A=\{A_0, \ldots, A_{n+1}\}$, where n is the total amount of activities. Activities A_0 and A_{n+1} are dummy activities and they represent the start and the end of the project respectively.

The duration of each activity is indicated by vector $p=\{p_0, \ldots, p_{n+1}\}$., where the duration of activity A_i is represented as p_i. The duration of dummy activities is $p_0 = p_{n+1} = 0$.

The precedence relationship of one task to another is represented by E, such that $(A_i, A_j) \in E$ means that activity A_j can only be executed after activity A_i has been completed. Precedence relationship can also be stated by the activity-on-node graph [41], in which nodes represent activities and transitions between nodes represent precedence relationships.

The resources are defined by a finite set $R=\{R_1, R_2, \ldots, R_q\}$ and the availability of each resource is represented as $B=\{B_1, B_2, \ldots, B_q\}$. The resource R_k is called unary or non-shareable if its availability is $B_k=1$. If the availability of resource is $R_k > 1$, the resource is regarded as shareable and can be occupied by several activities.

To represent the activities' demands for resources, the notation $\boldsymbol{b}$ is used. The amount of resource R_k per one time period during the execution of A_i is defined as b_{ik}.

The starting times of activities are abstracted by a schedule S, where S_i represents the start time of activity A_i. S_0 is used as a reference point. It signifies the start of the project and is always assumed to be 0. The total duration of the project, or makespan of a schedule, S will be equal to the start time of the last activity S_{n+1}.

Taking into consideration all formulation presented above, the optimization problem can then be stated as finding a non-pre-emptive schedule S of minimal makespan S_{n+1} (1) subject to resource (2) and precedence (3) constraints.

$$\text{Min:} \quad S_{n+1} \tag{1}$$

$$\text{Subject to:} \quad \sum_{Ai \in At} b_{ik} \leq B_k \quad \forall R_k \in \mathcal{R} \quad \forall t \geq 0 \tag{2}$$

$$S_j - S_i \geq p_i \quad \forall (A_i, A_j) \in E \tag{3}$$

The A_t in (2) represents a set of non-dummy activities that need to be schedules and can be calculated using (4).

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

$$A_t = \{A_i \in A \setminus S_i \leq t < S_i + p_i\} \quad (4)$$

III. Flower Pollination Algorithm

Flower Pollination Algorithm (FPA) is a novel nature-inspired metaheuristic algorithm based on the flower pollination process of flowering plants, which was created by Yang in 2012 [40].

Flower pollination process is typically associated with the reproduction of flowers, when flower pollen is transferred by various pollinators, such as insects, birds, and other animals. Flower pollination can be of two types: abiotic and biotic. About 80% of flowering plants belong to biotic pollination. This means that most of pollen is transferred by pollinators, like insects or animals. The rest 20% belong to abiotic and they can pollinate without any involvement of pollinators.

Some of pollinators are very diverse and they tend to visit only specific flower species. Such flower regularity can be regarded as evolutionary advantage, as it maximizes the transfer of the flower pollen to the same plants, therefore maximizing the reproduction of the flowers which belong to the same species.

Pollination can be achieved in two ways: self-pollination and cross-pollination. Cross-pollination refers to a process when a pollination occurs from a pollen of aflower of a different plant, while self-pollination is the fertilization of one flower from the pollen of the same species flower. Cross-pollination occurs at long distances, and is done by pollinators like bees and flies, which behave accordingly to Lévy flights behavior [42], with fly distance obeying a Lévy distribution. Moreover, flower constancy can be considered as an increment step using the similarity or difference between two flowers. According to Yang and Deb [43], in some optimization problems, the search for new solution is more efficient via Lévy Flights.

From the biological point of view, the main objectives of the flower pollination are the survival of the fittest and optimal reproduction of plants.

Based on the characteristics of the flower pollination process described above, Yang established the following rules for the FPA:

1) Biotic and cross-pollination processes are considered as global pollination process; Pollinators in this processes behave according to Lévy flights behavior;
2) Abiotic and self-pollination are considered as local pollination process;
3) Pollinators like insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved;
4) Switching between local and global pollinations is controlled by probability $p \in [0, 1]$.

With the rules outline above, the algorithm's pseudo-code can be formulated in Fig. 1.

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

```
Objective function: min/max f(x), x=(x1, ..., xd)
Initialize a population of n flowers
Find the best solution g in the population
Define a switch probability p
Define maxGeneration

while (generation < maxGeneration)
   for i = 1 : n
      if rand < p
         Global pollination via x_i^{t+1} = x_j^t + L(g - x_j^t), where L obeys Levy distribution
      else
         Choose two random flowers from population
         Local pollination
      end if
      evaluate new solutions
      If new solutions are better, add them to the population
   end for
   find g
   generation++
end while
```

Figure 1: Flower Pollination Algorithm pseudo-code

IV. Discrete Flower Pollination Algorithm for RCPSPs

In this paper, Discrete Flower Pollination Algorithm (DFPA) is proposed as a modification of the original FPA for solving combinatorial problems, such as RCPSPs. As the original FPA was designed for a continuous optimization problems, the concepts of such algorithm elements as flower, objective function, global pollination, Lévy Flights, and local pollination were changed.

A. Flower

In DFPA, a flower represents an individual in a population, which is presented in a form of permutation (Fig. 2), where each element is the scheduled activity and the index of the element is the order in which this activity is going to be executed. Each flower is considered as one solution. These permutations are positioned in the space according to the order of their components. The movement in the search space is accomplished by changing the order of the components and the length of step is derived from the value generate by Lévy flights. Movement can be done in three ways: small step, amount of small steps or large jump. To estimate the amount of steps and their length, the Lévy is calculated in an interval between 0 and 1, which then is used to derive the steps.

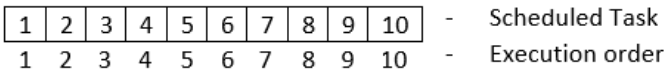

Figure 2: Solution representation

B. Objective Function

Objective function represents a numeric value which associates with the solution in the search space, therefore, the quality of the solution is evaluated by the makespan of the project.

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

C. Global Pollination

Changing the order of the tasks can be done in small or large steps. For a small step the swap mutation (Fig. 3) is used. With the swap mutation, the positions of two randomly selected tasks are switched respectively. To mimic a large step, the inverse mutation (Fig. 4) is used. With inverse mutation two tasks from a solution are selected randomly and all tasks in between them are swapped with places. Understandably, when swap and inverse mutations are performed, the precedence constrains must be satisfied.

Figure 3: Swap mutation example. A – Initial schedule, B – New schedule.

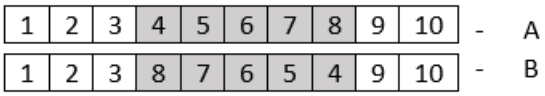

Figure 4: Inverse mutation example. A – Initial schedule, B – New schedule.

D. Lévy Flights

To improve the quality of the solutions, similarly to original FPA, the Lévy Flights (5) is used to calculate the length of the step.

$$Lévy(s, \lambda) \sim s^{-\lambda}, (1<\lambda<3) \quad (5)$$

Equation (5) has infinite variance with an infinite mean [42] and is used to derive the step size.

To make a choice between a small step, a number of small steps and a large step, the Lévy flights, associated with the interval between 0 and 1, is calculated. The steps are determined in the following way:

1) $[0,i]$ – move by one step (swap mutation);
2) $[(k\text{-}1) * i, k * i]$ – move by k amount of steps;
3) $[k * i, 1]$ - perform large jump (inverse mutation).

The value of i in this process is $(1 / (n+1))$, where n is the maximum amount of steps; and k is in $\{2, n\}$ region. For example, if $n = 4$, $i = 0.2$, the whole interval will be divided into the following five parts:

- Lévy in $[0, i] = [0, 0.2]$ – one small step;
- Lévy in $[i, i * 2] = [0.2, 0.4]$ – two small steps;
- Lévy in $[i*2, i* 3] = [0.4, 0.6]$ – three small steps;
- Lévy in $[i * 3, i * 4] = [0.6, 0.8]$ – four small steps;
- Lévy in $[i *4, 1] = [0.8, 1]$ – large step.

E. Local Pollination

The local pollination occurs via a crossover method, example of which is demonstrated in Fig. 5, where two randomly selected flowers from the population are combined into one. In this crossover method, a subset of tasks is

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

selected from the first flower and is used to create the new solution. Any missing tasks are then added to the new solution from the second flower in the same order they were found.

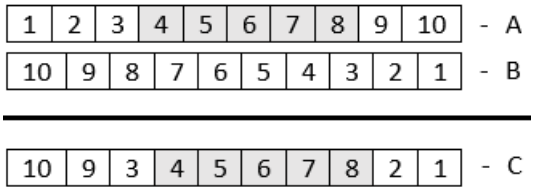

Figure 5: Local pollination Example. A – Flower 1, B – Flower 2, C – New flower.

VI. Experimental Results

A. Benchmark Problem

The performance and efficiency of the proposed algorithm are tested using the sets of RCPSP benchmark instances taken from the publicly available electronic library PSPLIB [44]. The PSPLIB consists of 2040 test projects with 30, 60, 90, and 120 activities, each project consisting of 4 limited resources, and each activity having a maximum of 3 successors. Due to the complexity of the RCPSP, the optimal makespan is only given for the projects with 30 activities, while optimal makespan of sets with 60 and more activities is still remains unknown. Therefore, to test the algorithm, only instances with 30 activities are considered. After all simulations are carried out, the DFPA is then compared with other recent heuristic methods which were used to solve RCPSP before, like Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization, Ant Colony Optimization and Priority Rule-based scheduling.

B. DFPA Parameter Settings Configuration

The DFPA has been implemented using Java programming language under a 64 bit Windows 8.1 operating system. All experiments were carried out on an Intel Core i7 2.4GHz laptop with 16GB of RAM.

The parameter settings (Table 1) for the DFPA were identified. Figure 6 demonstrates the impact of population sizes on the average value of all solutions found with the cases of maximum number of iterations of 25, 50 and 100, while Fig. 7 shows the effect of iterations with the same settings for the maximum number of iterations with the cases of population size of 5, 25 and 50. The experiment results, presented on Fig. 6 and Fig. 7, were received from the execution of the j3039_3 PSPLIB instance, which has the optimal makespan of 54. Bigger population sizes and higher maximum iterations let the algorithm to find better solutions, however, this also results in higher computational time.

TABLE 1
DFPA PARAMETER SETTINGS

Parameter	Value	Comment
n	20	Population size
p	0.8	Switch probability
MaxGeneration	1000	Maximum number of iteration

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

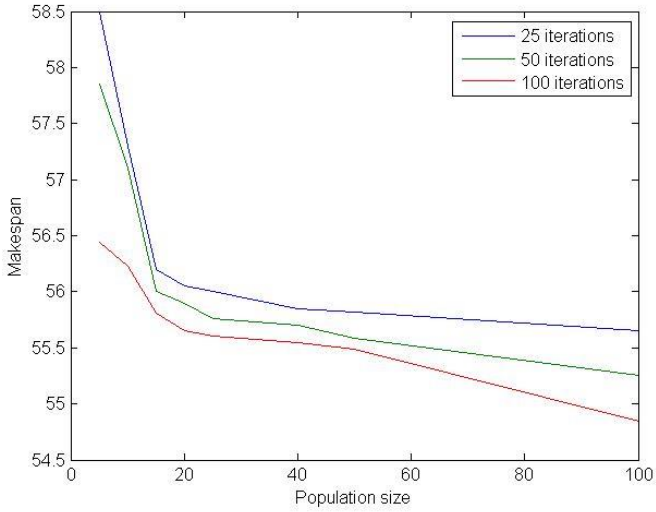

Figure 6: Dependency of average duration of best solution from population size for j3039_3 benchmark instance set

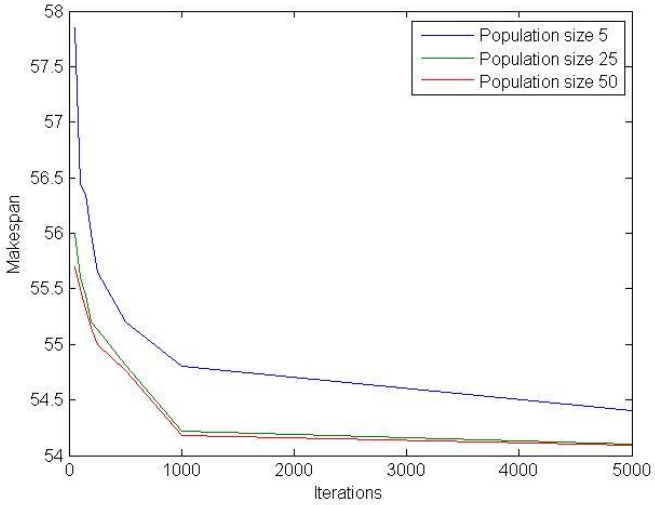

Figure 7: Dependency of average duration of best solution from maximum amount of iterations for j3039_3 benchmark instance set

C. Performance Evaluation

To test the algorithm in each case 100 independent runs with each benchmark instance set have been performed. The selected benchmark instances, presented in Table 2, were chosen randomly from the total amount of 480 sets. The results of the experiments are summarized in Table 2, where the first column shows the name of the instance set, the optimal makespan of the benchmark instance set taken from PSPLIB is displayed in the second column. The column "best" shows the makespan of the best solution found by the DFPA, similarly, the column "worst" shows the makespan of the worst solution. The column "average" contains the average project duration based on the 100 runs of each set. The column "Dev (%)" denotes the percentage deviation of the average solution makespan from the optimal solution makespan and is calculated using (6).

$$Dev\ (\%) = (solution\ makespan - optimal\ solution\ makespan) / optimal\ solution\ makespan * 100 \quad (6)$$

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

TABLE 2
COMPUTATIONAL RESULTS OF DFPA SIMULATIONS

Instance name	Optimal Solution	Best	Worst	Average	Dev (%)	Time (s)
j3006_02	51	51	59	51.54	1.06	1.83
j3015_04	48	48	52	48.14	0.29	0.40
j3020_01	57	57	59	57.14	0.24	0.40
j3026_06	53	53	56	53.18	0.34	0.51
j3029_04	103	103	110	103.48	0.47	2.01
j3034_04	67	67	74	67.28	0.42	0.30
j3039_03	54	54	57	54.12	0.22	0.40
j3042_08	82	82	83	82.34	0.41	0.53
j3045_02	125	125	132	125.70	0.56	0.68
j3048_02	54	54	58	54.18	0.33	0.42
Average					**0.434**	

Based on the results from Table 2, it can be concluded that DFPA was capable of finding the optimal solutions for all chosen benchmark instances and the average deviation percentages from optimal solution based on 100 runs in all cases is less than 1.06%. These results, presented in Table 2, indicate that DFPA is indeed powerful algorithm and can provide adequate solutions in reasonable time.

D. Comparison with Other Algorithms

Lastly, in Table 3, the experimental results of the DFPA are compared with other heuristic algorithms, results of which are taken from [2, 36, 45]. The numbers 1000 and 5000 in Dev (%) column denote the maximum number of iterations and are used as a stop criterion. The algorithms presented in Table 3 were selected based on their complexity. Only original non-hybrid versions of algorithms were chosen and modified versions of metaheuristic algorithms with additional more complicated search mechanisms, e.g. radius PSO [31] or random key-based GA [20], were omitted and left out.

TABLE 3
COMPARISON OF PERFORMANCE OF OTHER ALGORITHMS

Algorithm name	Author(s)	Dev (%)	
		1000	5000
DFPA	**This paper**	**0.434**	**0.21**
ACO [36]	Luo, Wang	0.39	0.22
SA [28]	Bouleimen, Lecocq	0.38	0.23
GA [6]	Hartmann	0.54	0.25
PSO [31]	Anantathanvit, Munlin	0.41	0.33
Tabu Search [46]	Baar et al.	0.86	0.44
Adaptive sampling [47]	Kolisch	0.74	0.52
Serial sampling LFT [47]	Kolisch	0.83	0.53
Serial random sampling [48]	Schrimer, Riesenberg	0.71	0.59
Parallel sampling WCS [47]	Kolisch	1.40	1.28
Parallel sampling LFT [47]	Kolisch	1.40	1.29

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

Overall, the comparison of the performance with other algorithms can be regarded as satisfactory and it can be noted that DFPA has managed to outperform all algorithms presented in Table 3. Better performance of DFPA over other algorithms can be explained with a good balance between exploitations and exploration, intelligent use of Lévy Flights and the reduced number of parameters that need to be configured to provide the optimal performance. Another DFPA's advantage, which plays an important role in deciding which algorithm is better, is its simplicity. Being very simple, the DFPA is easy to implement, which makes it more attractive to be used in other combinatorial problems.

VII. Conclusions

In this paper, a new metaheuristic FPA is selected and then modified for solving the combinatorial optimization problems. As the original FPA was designed for solving a continuous optimization problems, in order to adapt it for solving combinatorial problems, the concepts of such algorithm elements as flower, objective function, global pollination, Lévy Flights, and local pollination were changed. Further, the algorithm's performance has been tested on a set of PSPLIB benchmark instances, and despite being simple and relatively easy to implement, the proposed algorithm has managed to find optimal solutions in all benchmark instances and its average deviation from the optima based on 100 runs in all cases was less than 1.06%, which has validated algorithm's effectiveness. Lastly, the algorithm has been compared with other popular metaheuristic non-hybrid algorithms, like GA, SA, PSO, and ACO and the results of comparison have shown that DFPA has managed to outperform all selected algorithms in terms of average deviation percentage from the optimal solution, therefore proving its competitiveness and superiority over selected algorithms for comparison. These results indicate that despite being very simple, the DFPA is yet very powerful and efficient algorithm.

In the future, the work on improvement of DFPA will be carried on and the algorithm will be applied in solving more complicated scheduling problems. The probable areas of further application will include traveling salesman problem and knapsack problem. One of the possible areas of improvement is the better exploitation of global solution to make the chance of falling in local trap even less than it is now. Further, after this improvement is done, it will be compared with genetic algorithm with the aim of finding which algorithm finds the global solution more efficiently.

REFERENCES

[1] J. Blazewicz, J. Lenstra and K. Rinnooy, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Applied Mathematics*, vol. 5, no. 10, pp. 11-24, 1983.
[2] B. Chen and G. Quan, "NP-Hard Problems of Learning from Examples," in *Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Shandong, 2008.
[3] R. Kolisch and S. Hartmann, Handbook on Recent Advances in Project Scheduling, Kluwer: Dordrecht, 1999.
[4] J. Stinson and W. Davis, "Multiple Resource-Constrained Scheduling Using Branch and Bound," *AIIE Transactions*, vol. 10, no. 3, pp. 252-259, 1978.
[5] W. Iskander and S. Panwalker, "A survey of scheduling rules," *Operations Research*, vol. 77, no. 2, pp. 81-98, 1979.
[6] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling problem," *Naval Research Logistics*, vol. 45, no. 10, p. 733–750, 1998.
[7] H. Zhang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," *Automation in Construction*, vol. 14, no. 3, p. 393–404, 2005.
[8] W. Zhang, Y. Huang and M. Wang, "Synthetic Optimization in Project Schedule by Using Ant Colony Algorithm," in *World Congress on Intelligent Control and Automation*, Dalian, 2010.
[9] L. Lawler and E. Wood, "Branch and Bound Methods: A Survey," *Operations*, vol. 14, no. 4, pp. 699-719, 1966.

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

[10] T. Johnson, An algorithm for the resource-constrained project scheduling problem., Cambridge, MA: Doctoral Thesis, Massachusetts Institute of Technology, 1967.
[11] H. Muller, "A method for planning the optimum input use," *The journal of economical production,* vol. 64, no. 4, pp. 135-140, 1967.
[12] A. Sprecher and A. Drexl, Manuscripts from the institutes of Business Administration, Kiel: University of Kiel, 1996.
[13] E. Davis and J. Patterson, "A Comparison of Heuristic and Optimum Solutions in Resource- Constrained Project Scheduling," *Management Science S2: Application Series,* vol. 21, no. 8, p. 944–955, 1975.
[14] T. R. Browning and A. A. Yassin, "Resource-constrained multi-project scheduling: Priority rule performance revisited," *International Journal of Production Economics,* vol. 126, no. 2, p. 212–228, 2010.
[15] S. R. Lawrence and T. E. Morton, "Resource-constrained multi-project scheduling with tardy cost: Comparing myopic, bottleneck, and resource pricing heuristics," *European Journal of Operational Research,* vol. 64, no. 2, pp. 168-187, 1993.
[16] D. W. Hildum, Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems., University of Massachusetts, Amberst, MA: Ph.D. Dissertation, Computer Science Dept., 1994.
[17] F. F. Boctor, "Heuristics for scheduling projects with resource restrictions and several resource-duration modes," *International Journal of Production Research,* vol. 31, no. 11, pp. 2547-2558, 1992.
[18] P. Husbands, G. Germy and M. McIlhagga, "Two applications of genetic algorithms," in *AISB workshop on evolutionary computing*, Chicago, CH, 1996.
[19] L. Davis, "Hybrid genetic algorithms for machine learning," in *IEEE Colloquium on Machine Learning*, London, 1990.
[20] J. J. Mendes, "A random key based genetic algorithm for the resource constrained project scheduling problem," *Computers & Operations Research*, no. 36, p. 92–109, 2009.
[21] J. R. Montoya-Torres and E. Gutierrez-Franco, "Project scheduling with limited resources using a genetic algorithm," *International Journal of Project Management,* vol. 28, no. 6, p. 619–628, 2009.
[22] M. Shahsavar and S. T. AkhavanNiaki, "An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus–penalty policy in resource investment problem," *Advances in Engineering Software*, vol. 41, p. 1023–1030., 2010.
[23] J. Alcaraz, C. Maroto and R. Ruiz, "Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with genetic algorithms," *Journal of the Operational Research Society,* vol. 54, p. 614–626, 2003.
[24] P. J. M. v. Laarhoven, E. H. L. Aarts and J. K. Lenstra, "Job Shop Scheduling by Simulated Annealing," *Operations Research,* vol. 40, no. 1, pp. 113-125, 1992.
[25] G. Palmer, "A simulated annealing approach to integrated production scheduling," *Journal of Intelligent Manufacturing,* vol. 7, no. 3, pp. 163-176, 1996.
[26] F. F. Boctor, "Resource-constrained project scheduling by simulated annealing," *International Journal of Production Research,* vol. 34, no. 8, pp. 2335-2351, 1996.
[27] Y. Nikulin and A. Drexl, "Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models," *Journal of Scheduling,* vol. 13, pp. 261-281, 2010.
[28] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research,* vol. 26, no. 149, p. 268–281, 2003.
[29] R. Zamani, "An efficient time-windowing procedure for scheduling projects under multiple resource constraints," *OR Spectrum,* vol. 26, no. 3, pp. 423-440, 2004.
[30] H. Zhang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," *Automation in Construction,* vol. 14, no. 3, p. 393–404, 2005.
[31] M. Anantathanvit and M. Munlin, "Radius particle swarm optimization for resource constrained project scheduling problem," in *International Conference on Computer and Information Technology*, Khulna, 2014.
[32] M. Li, Z. Yuanbiao, W. Jiang and J. Xie, "A Particle Swarm Optimization Algorithm with Crossover for Resource Constrained Project Scheduling Problem," in *IITA International Conference on Services Science, Management and Engineering*, Zhangjiajie, 2009.
[33] D. Linyi and Y. Lin, "International Conference on Computational Intelligence and Security," in *International Conference on Computational Intelligence and Security*, Harbin, 2007.
[34] K. Zhang, G. Zhao and J. Jiang, "Particle swarm optimization method for resource-constrained project scheduling problem," in *International Conference on Electronic Measurement & Instruments*, Beijing, 2009.
[35] D. Merkle, H. Schmeck and M. Middendorf, "Ant colony optimization for resource-constrained project scheduling," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 4, pp. 333 - 346, 2002.
[36] S. Luo, C. Wang and J. Wang, "Ant colony optimization for resource-constrained project scheduling with generalized precedence relations," in *International Conference on Tools with Artificial Intelligence*, Beijing, 2003.
[37] J. Wang, S. Zhang and J. Chen, "Resource-constrained multi-project scheduling based on ant colony optimization algorithm," in *Intelligent Computing and Intelligent Systems*, Xiamen, 2010.
[38] Y. Shou, "A Bi-directional Ant colony algorithm for resource constrained project scheduling," in *Industrial Engineering and Engineering Management*, Singapore, 2007.
[39] S. T. Lo, R.-M. Chen, Y.-M. Huang and C.-L. Wu, "Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system," *Expert Systems with Applications: An International Journal,* vol. 34, no. 3, pp. 2071-2081 , 2008.
[40] X.-S. Yang, "Flower Pollination Algorithm for Global Optimization," *Unconventional Computation and Natural Computation,* vol. 7445, pp. 240-249, 2012.
[41] Y. Zhou and Y. Chen, "Business process assignment optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, Paris, 2002.
[42] M. F. Shlesinger, G. M. Zaslavsky and U. Frisch, "Lévy Flights and Related Topics in Physics," in *Proceedings of the International Workshop*, Nice, France, 1994.
[43] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy flights," in *World Congress on Nature & Biologically Inspired Computing*, Coimbatore, 2009.
[44] PSPLIB, "Project Scheduling Problem LIBrary," [Online]. Available: http://129.187.106.231/psplib/. [Accessed 6 May 2015].
[45] Y. Zhou and Y. Chen, "Business process assignment optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, Paris, 2002.

[46] T. Baar, P. Brucker and S. Knust, "Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem," in *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimizatio*, Springer US, 1999, pp. 1-18.

[47] R. Kolisch and S. Hartmann, "Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis," *International Series in Operations Research & Management Science*, vol. 14, pp. 147-178, 1999.

[48] A. Schirmer and S. Riesenberg, "Parameterized Heuristics for Project Scheduling - Biased Random Sampling Methods," *Manuskripte aus den Instituten fur Betriebswirtschaftshehre der Universitaten Kiel*, vol. 471, pp. 129-141, 1997.

http://sites.google.com/site/ijcsis/
ISSN 1947-5500